Transformers in NLP using ALBERT and Distill BERT

Business Objective

In Part-1 of the Transformer series, we have discussed the techniques of NLP, starting from Bag of words to Transformer architecture. We finally discussed BERT which is one of the State-of-the-Art Transformer models for downstream NLP tasks (Multi-Class Text Classification with Deep Learning using BERT)

In Part-2 of the series, we came to know the limitations of BERT and the ways to improve it. We then explored the concepts of Auto Regression, Auto Encoding and two new models, RoBERTa and XLNet which further improved on BERT performance significantly by changes in training techniques and architectures. (Text Classification with Transformers-RoBERTa and XLNet Model)

In Part-3 of this series, we will deal with the short comings of all these models in terms of Memory Optimization, Prediction Latency & Space usage. We will understand new techniques and architecture modifications that help to solve these issues while deploying a model to production and study in detail two new models, which are:

- ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
- DistilBERT: A distilled version of BERT: smaller, faster, cheaper, and lighter

And see how these optimize space and memory with minor changes in prediction accuracy.

Finally, we will have a comparative study across all the five transformer models over the three series.

Data Description

The DBpedia ontology classification dataset is constructed by picking 14 non-overlapping classes from DBpedia 2014. They are listed in classes.txt. From each of these 14 ontology classes, we randomly choose,

- 40,000 training samples and 5,000 testing samples.
- Therefore, the total size of the training dataset is 560,000 and the testing dataset 70,000.

There are three columns in the dataset,

- Title: the title of the document
- Content: the body of the document
- Label: one of the 14 possible topics.

Aim

The project aims at building two models, namely ALBERT and DistilBERT to perform classification on the DBpedia ontology dataset.

Tech stack

- Language - Python
- Libraries -datasets, numpy, pandas, matplotlib, ktrain, transformers, tensorflow, sklearn

Environment

- Jupyter Notebook
- Google Colab Pro (Recommended)

Approach

1. Install the required libraries
2. Load the 'DBpedia' dataset
3. Load train test data
4. Data pre-processing
    - Assign column names to the dataset
    - Append and save the dataset
    - Drop redundant columns
    - Add text length column for visualization
5. Perform data visualization
    - Histogram plots
6. ALBERT model
    - Check for hardware and RAM availability S
    - Import necessary libraries
    - Data interpretations
    - Create an ALBERT model instance.
    - Split the train and validation data
    - Perform Data Pre-processing
    - Compile ALBERT model in a K-train learner object
    - Fine-tune the ALBERT model on the dataset
    - Check for the performance on validation data
    - Save the ALBERT model

7. DistilBERT model
    - Check for hardware and RAM availability
    - Import necessary libraries
    - Data interpretations
    - Create a DistilBERT model instance.
    - Split the train and validation data
    - Perform Data Pre-processing
    - Compile DistilBERT model in a K-train learner object
    - Fine-tune the DistilBERT model on the dataset
    - Check for the performance on validation data
    - Save the DistilBERT model
8. Similarly create a BERT model using the DBpedia dataset for comparative study
9. Follow the above steps for creating a BERT model on the 'Emotion' dataset
10. Follow the above steps for creating an ALBERT model on the 'Emotion' dataset
11. Follow the above steps for creating a DistilBERT model on the 'Emotion' dataset
12. Save all the generated model

Modular code overview

```
input
  |_dbpedia_14_test.csv
  |_dbpedia_14_train.csv
  |_dbpedia_csv.tar.gz

src
  |_Engine.py
  |_ML_Pipeline
            |_albert.py
            |_distilbert.py
            |_feature_engineering.py
            |_model.py
            |_utils.py


lib
  |_AlBert_on_Dbpedia_14.ipynb
  |_AlBert_on_Emotion.ipynb
  |_Bert_on_Dbpedia_14.ipynb
  |_Bert_on_Emotion.ipynb
  |_Dbpedia_Ontology_Dataset_Visualization.ipynb
  |_DistilBert_on_Dbpedia_14ipynb
  |_DistilBert_on_Emotion.ipynb
  |_Transformers_in_NLP_Part_3_Albert_DistilBert.ipynb

output
  |_albert-content
  |_distilbert-content
```

Once you unzip the modular_code.zip file you can find the following folders within it.

1. Input
2. Src
3. Output
4. Lib

1. Input folder - It contains all the data that we have for analysis. There are two csv files in our case along with a zip file
   - dbpedia_14_test
   - dbpedia_14_train
   - dbpedia_csv.tar.gz

2. Src folder - This is the most important folder of the project. This folder contains all the modularized code for all the above steps in a modularized manner. This folder consists of:
   - Engine.py
   - ML_Pipeline
     The ML_Pipeline is a folder that contains all the functions put into different python files which are appropriately named. These python functions are then called inside the Engine.py file.

3. Output folder - The output folder contains the albert and distilbert models that we trained for this data. This model can be easily loaded and used for future use and the user need not have to train all the models from the beginning.
   Note: This model is built over a chunk of data. One can obtain the model for the entire data by running Engine.py by taking the entire data to train the models.

4. Lib folder - This folder contains the original Ipython notebooks that we saw in the videos. There are around 8 notebooks, which were seen during the videos.

Project Takeaways

1. Understanding the business problem.
2. BERT model overview
3. Understanding of the ALBERT model (A Lite BERT for Self-supervised Learning of Language Representations)
4. Understand Factorized Embedding Parametrization, Cross Layer Parameter Sharing, and Sentence Order Prediction
5. Import the data from the hugging face library.
6. Perform data pre-processing on the data.
7. Build an ALBERT model instance, compile and fine-tune the model
8. Understanding of the DistilBERT model
9. Concept of Knowledge Distillation

10. Build a DistilBERT model instance, compile and fine-tune the model
11. Evaluate the models based on performance metrics
12. Evaluate the models on unseen data (test data)
13. Save the models
14. Create the BERT, ALBERT, and DistilBERT models on a different dataset
15. A comparative study across multiple models.